Bounding the Resource Availability of Activities with Linear Resource Impact

Jeremy Frank and **Paul H. Morris**
Computational Sciences Division
NASA Ames Research Center, MS 269-3
frank@email.arc.nasa.gov
Moffett Field, CA 94035

Abstract

We introduce the Linear Resource Temporal Network (LRTN), which consists of activities that consume or produce a resource, subject to absolute and relative metric temporal constraints; production and consumption is restricted to linear functions of activity duration. We show how to construct tight bounds for resource availability as a function of time in LRTNs; for a broad class of problems, the complexity is polynomial time in the time horizon h and the number of activities n. Our approach extends a maximum flow formulation previously used to construct tight bounds for networks of events with instantaneous resource impact. We construct the bounds in two parts: we make direct use of maximum flows to find local maxima of the upper bound (and symmetrically local minima of the lower bound), and we solve a related linear programming problem to find local minima of the upper bound (and symmetrically local maxima of the lower bound).

Introduction

Scheduling requires creating an ordering for tasks that satisfies temporal and resource constraints. Building schedules by ordering events rather than assigning event times preserves temporal flexibility; this permits the construction of a family of schedules without determining exactly when events take place while still guaranteeing that feasible solutions exist. Preserving flexibility has two potential advantages over finding a "ground" schedule. The first advantage is protection from uncertainty that can lead to costly rescheduling during schedule execution. Usually, scheduling is performed assuming that the problem's characteristics are known in advance, do not change, and that the execution of the schedule is deterministic. These assumptions are often violated in practice; for example, activity duration may not be precisely known. The creation of temporally flexible plans to protect against some execution time uncertainty was described in (Morris, Muscettola, & Tsamardinos 1998) and was successfully used in controlling a spacecraft (Jónsson *et al.* 2000). The second advantage is in speeding up search for feasible schedules. In this context, the goal is to reduce the search space from the set of all start and end times of activities to the set of all activity ordering decisions. This approach to scheduling was studied in (Cheng & Smith 1995), (Laborie 2003a) and (Policella *et al.* 2004).

 This work was funded by the NASA Exploration Technology Development Program.

Laborie (Laborie 2003b) describes a simple but expressive formalism for scheduling problems called *Resource Temporal Networks* (RTNs). Briefly, RTNs consist of a *Simple Temporal Network* (STN) as described in (Dechter, Meiri, & Pearl 1991), constant instantaneous resource impacts (either production or consumption) for each timepoint, and piecewise constant resource bounds. Instantaneous impacts are useful for modelling reusable resources that are allocated at the beginning of an activity and released at the end, such as power usage on a planetary rover. Three techniques have been developed to bound the resource availability for RTNs in polynomial time; the Balance Constraint (Laborie 2003a), the Resource Envelope (Muscettola 2002), and the Flow Balance Constraint (Frank 2004). These bounds can be used to terminate the process of generating temporally flexible schedules. In particular, both (Muscettola 2002) and (Frank 2004) provide bounds that are tight, in the sense that they justify the resource bound by proving the existence of a feasible schedule. These latter techniques can be used as sound and complete reasoning in search.

Many practical problems have activities that gradually consume or replenish some resource like energy or data. In this paper, we generalize the RTN problem class to Linear Resource Temporal Networks (LRTNs), where *activities* consist of a start, end and duration, the resource impact of activities is permitted to be linear in the duration of the activity, and the resource bounds can be piecewise linear functions rather than piecewise constants. We show how to construct tight bounds for resource availability as a function of time in LRTNs; for a broad class of LRTNs, the complexity is polynomial time in the time horizon h and the number of activities n. Our approach makes use of the maximum flow formulation described in (Muscettola 2002). We identify some properties LRTNs must have in order for the maximum flow formulation to apply, then show how to reformulate LRTNs so that these properties hold. We make direct use of maximum flows to find local maxima of the upper bound (and symmetrically local minima of the lower bound). We then adapt this approach and solve a related linear programming problem to find local minima of the upper bound (and symmetrically local maxima of the lower bound).

Notation and Definitions

We will assume LRTNs have a single resource. We will also assume a constant resource upper bound R_{ub} and a lower bound of 0, and that the resource initially has R_{ub} available capacity. This easily generalizes to varying initial capacity, piecewise constant upper bounds, and (with some additional work) to piecewise linear upper and lower bounds.

Let $\mathcal{A}$ be the set of all activities of an LRTN and $n = |\mathcal{A}|$. Let $A \in \mathcal{A}$ be an activity. Let A_s be the start event of activity A, and let A_e be the end event of A. If G is a ground schedule, A_s^G denotes the value of A_s in G, and similarly for A_e^G. Activity durations and resource rates are denoted by A_d and A_r, respectively. If $A_r < 0$, then A is said to be a *consumer*; if $A_r > 0$, then A is a *producer*.

Each activity A has associated with it a constraint:

$$A_s + A_d = A_e$$

Let $\mathcal{E}$ be the set of timepoints (start or end times of activities) and suppose $E_1, E_2 \in \mathcal{E}$. There may be many *simple temporal constraints* of the form

$$x_1 \leq E_1 - E_2 \leq x_2$$

We let the "dummy" activity H indicate the scheduling horizon, thus $H_s = 0$, $H_e = h$ and (obviously) $H_d = h$. Absolute constraints on events are then translated into simple temporal constraints between events and H_s or H_e. (For example, the constraint $A_s \in [x_1, x_2]$ translates to $x_1 \leq A_s - H_s \leq x_2$.) Recall that an STN can be transformed into a *distance graph* (by expressing all the constraints as upper bound constraints). For a consistent STN, we denote the shortest path distance from a timepoint E_1 to a timepoint E_2 in the distance graph by $d(E_1, E_2)$. This provides an upper-bound on the temporal distance from E_1 to E_2; thus, $E_2^G - E_1^G \leq d(E_1, E_2)$ in every grounded schedule G. Note that $d(H_s, E)$ and $-d(E, H_s)$ provide absolute upper and lower bounds on E; we denote these by E_{ub} and E_{lb}, respectively. An STN may be regarded as a concise representation of a flexible schedule.

Given a ground schedule G, we denote by $Avail_G(t)$ the available resource at t in G:

$$Avail_G(t) = \sum_{A \in \mathcal{A} | A_e^G \leq t} A_r A_d + \sum_{A \in \mathcal{A} | A_s^G \leq t < A_e^G} (t - A_s^G) A_r$$

.

We denote by $L_{max}(t)$ the *maximum available* resource at a time t over all schedules, and by $L_{min}(t)$ the minimum available resource at t. Thus, $L_{max}(t) = \max_G Avail_G(t)$ and similarly for $L_{min}(t)$. We say G justifies $L_{max}(t)$ if $L_{max}(t) = Avail_G(t)$. We denote $\max_t L_{max}(t)$ by L_{max}^+ and $\min_t L_{max}(t)$ by L_{max}^- (similarly for L_{min}^+ and L_{min}^-).

Note that L_{max} and L_{min} are functions that tightly bound the availability of the schedules; we call these the upper and lower envelopes, respectively, following (Muscettola 2002).

The results of this paper are limited to the case where the activity durations and resource consumption rates are constant and provided as inputs to problem instances; to emphasize this, we will henceforth denote them by a_d and a_r for an activity A.

The theory of maximum flows is described in standard textbooks, for example (Ahuja, Magnanti, & Orlin 1993). Let F be a flow graph and *flow* a flow; we will denote pipes of the flow graph by p. As in (Muscettola 2002), we will denote the *residual capacity* of a *flow* over a subset of the flow graph $G \subset F$ by $r_{flow}(G)$. That is, $r_{flow}(G)$ denotes the cumulative remaining capacity of pipes $p \in G$ reachable from the source for the maximum flow.

We introduce the following definitions:

Definition 1 *Given an LRTN, we define the* instants I *as* $\bigcup_{E \in \mathcal{E}} (\{E_{lb}\} \cup \{E_{ub}\})$.

Definition 2 *Given an LRTN, at time* t *an event* $E \in \mathcal{E}$ *is* pending *at* t *if* $E_{lb} \leq t \leq E_{ub}$, open *at* t *if* $t \leq E_{lb}$ *and* closed *at* t *if* $E_{ub} \leq t$. *An activity* A *is* closed *if* A_e *is closed,* open *if* A_s *is open,* completely pending *if* A_s *and* A_e *are pending, and* partially pending *otherwise.*[1]

Definition 3 *Given an LRTN and a pair of activities* A *and* B, A anti-precedes B *if* $d(A_e, B_e) \leq 0$.[2]

Definition 4 *Given an LRTN, a set of activities* $\mathcal{S}$ *is a* predecessor set *if, when activity* $S \in \mathcal{S}$, *then every activity* T *that* S *anti-precedes is also in* $\mathcal{S}$.

Definition 5 *Given an LRTN and a ground schedule* G, G *is* split *at* t *if there is some activity* A *such that* $A_s^G < t < A_e^G$. *A schedule that is not split at* t *is* intact *at* t.

Finding the Envelope of an RTN in Polynomial Time

Both RTNs and LRTNs can be solved using chronological search over ordering decisions. At any point in chronological search, if $L_{max}^- < 0$ or $L_{min}^+ > R_{ub}$, or any temporal constraints are violated, then an unavoidable conflict has been introduced and backtracking is necessary. If $L_{min}^- < 0$ or $L_{max}^+ > R_{ub}$ then at least one schedule leading to a violation of the resource bounds exists, so at least one ordering decision is necessary to eliminate this schedule. Finally, if $L_{min}^- \geq 0$ and $L_{max}^+ \leq R_{ub}$, then *all* schedules consistent with the temporal constraints are also consistent with the resource constraints. The resulting RTN or LRTN represents a *family* of schedules, or a temporally flexible schedule.

In this section we review the algorithm used in (Muscettola 2002) to find L_{max} for RTNs in polynomial time, which involves a maximum flow formulation. Maximum flow theory is useful for solving problems that match reward and shared costs in an optimum way. This can be exploited in scheduling where the "costs" of production events are the consumption events that necessarily precede them.

Suppose we are given an RTN with $c(X)$ denoting the resource impact of event X. To find the value of the schedule justifying $L_{max}(t)$, a maximum flow problem is constructed using all anti-precedence links derived from the arc-consistent STN. The rules for building the flow problem to

[1] Note an event may be both pending and open/closed, but an activity with duration > 0 is in only one state.

[2] Implies $B_e - A_e \leq 0$, i.e., every part of B non-strictly precedes some part of A, in every schedule.

find $L_{max}(t)$ are as follows: all pending events are represented by nodes of the flow problem. If $d(X, Y) \leq 0$ then the flow problem contains an arc $X \rightarrow Y$ with infinite capacity. If $c(X) > 0$ then the problem contains an arc $\sigma \rightarrow X$ with capacity $c(X)$. If $c(X) < 0$ then the problem contains an arc $X \rightarrow \tau$ with capacity $|c(X)|$. (The flow problem for $L_{min}(t)$ is constructed similarly, except if $c(X) > 0$ then the problem contains an arc $X \rightarrow \tau$ with capacity $c(X)$, and if $c(X) < 0$ then the problem contains an arc $\sigma \rightarrow X$ with capacity $|c(X)|$.) The maximum flow of this flow network matches all possible production with all possible consumption in a manner consistent with the precedence constraints. An RTN and associated flow problems for finding $L_{max}(t)$ and $L_{min}(t)$ are shown in Figure 1. The set of events reachable in the residual flow network is a predecessor set (since the infinite pipes resulting from the anti-precedes relation always have residual capacity), and is denoted $P_{max}(t)$. $L_{max}(t)$ is justified by scheduling all pending events in $P_{max}(t)$ before t, and all other pending events after t. The tightness of the bound is guaranteed by proving that adding the constraints that force these activities to occur before or after t is consistent with the original STN.

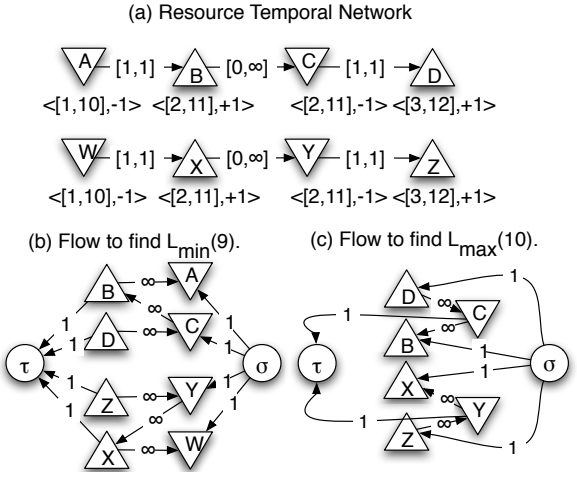

Figure 1: An RTN, the flow problem to find $L_{min}(9)$, and the flow problem to find $L_{max}(10)$.

In order to find L_{max}, (Muscettola 2002) shows that it is sufficient to find $L_{max}(t)$ at the instants I, since the envelope is constant between instants. The maximum flow problem can be solved using many well-known polynomial time algorithms; for RTNs with n events, thus, the complexity of finding L_{max} and L_{min} is polynomial.

Bounding Above the Envelope of an LRTN in Polynomial Time

We now show how to extend the use of the above maximum flow formulation for RTNs in order to bound above L_{max} for LRTNs in polynomial time. Our approach is to chop the activities into unit pieces, retain the linear resource impact, and use the flow formulation to find $L_{max}(t)$ at critical times, provided the constraints are integer-valued. The key to this approach is to focus attention on intact schedules at the instants.

Proceeding along these lines, assume that activity resource impact rates a_r are provided as inputs, all activity durations $a_d = 1$, and all temporal constraint bounds have integer values. (Decimal bounds with a fixed number of decimal places, which are sufficient for many practical problems, can be reduced to integer bounds by a simple transformation of units.) Resource rates may be arbitrary real numbers. We assume that the simple temporal constraints of the LRTN are consistent. Then the following properties hold:

1. The instants I are a (possibly proper) subset of the set of integers $1..h$.
2. At any instant, any activity is either open, closed or completely pending.
3. For every pair of timepoints E_1, E_2 the quantity $d(E_1, E_2)$ is an integer. Further, $d(A_e, B_s) < 0 \Rightarrow d(A_s, B_s) \leq 0 \Rightarrow d(A_e, B_e) \leq 0$.
4. It follows that for every pair of activities A, B, either
   (a) A anti-precedes B, i.e., $d(A_e, B_e) \leq 0$.
   (b) B can follow A, i.e., $d(A_e, B_s) \geq 0$.

We now prove two fundamental results that will allow us to design an algorithm for bounding above L_{max}.

Theorem 1 *Consider a consistent LRTN with integer temporal constraints and unit durations ($a_d = 1$). Then for every predecessor set $\mathcal{S}$, we have $\max_{A \in \mathcal{S}} A_{elb} \leq \min_{B \notin \mathcal{S}} B_{sub}$. Moreover, for all times t such that $\max_{A \in \mathcal{S}} A_{elb} \leq t \leq \min_{B \notin \mathcal{S}} B_{sub}$, there is a consistent schedule G intact at t such that every activity $A \in \mathcal{S}$ has $A_e^G \leq t$ and every activity $B \notin \mathcal{S}$ has $t \leq B_s^G$.*

Proof: Let $\mathcal{S}$ be a predecessor set. Suppose $\max_{A \in \mathcal{S}} A_{elb} > \min_{B \notin \mathcal{S}} B_{sub}$. Then there are activities $A \in \mathcal{S}$ and $B \notin \mathcal{S}$ such that $A_{elb} > B_{sub}$. It follows that $d(A_e, B_s) < 0$. By property 4, this implies A anti-precedes B. Since $\mathcal{S}$ is a predecessor set, that means B is in $\mathcal{S}$, which is a contradiction. Thus, $\max_{A \in \mathcal{S}} A_{elb} \leq \min_{B \notin \mathcal{S}} B_{sub}$.

Now let t be any time such that $\max_{A \in \mathcal{S}} A_{elb} \leq t \leq \min_{B \notin \mathcal{S}} B_{sub}$. If there are no completely pending activities at t the result follows trivially, so assume there is some completely pending activity. Suppose we add new constraints that force all activities in $\mathcal{S}$ to end before t, i.e. $\forall A \in \mathcal{S}$ add the constraint $A_e \leq t$, and force all the activies not in $\mathcal{S}$ to start after t, i.e. $\forall B \notin \mathcal{S}$ add the constraint $t \leq B_s$. Assuming the resulting STN is consistent, these constraints are sufficient to ensure that there is a consistent schedule G such that every activity in $A \in \mathcal{S}$ has $A_e^G \leq t$ and every activity $B \notin \mathcal{S}$ has $t \leq B_s^G$, and hence G is intact at t.

Suppose the resulting STN is not consistent. Then there is some simple (no repetitions) negative cycle that involves the new constraints. Note that the negative cycle must involve at least two of the added constraints, since each of the new constraints is individually consistent with the pre-existing network. Moreover, since the negative cycle has no repeated

nodes, and each of the added constraints involves t, the nogood must include exactly two of the new constraints, one added upper bound (from an activity A in $\mathcal{S}$) and one added lower bound (from an activity B not in $\mathcal{S}$), i.e, the nogood includes $A_e \overset{0}{\leftarrow} t \overset{0}{\leftarrow} B_s$. Since this portion of the nogood adds up to 0, the rest of the nogood (the path from A_e to B_s in the original network) must be negative. By property 4, this is only possible if A anti-precedes B, which contradicts the fact that B is not in $\mathcal{S}$. □

Corollary 1.1 *For every predecessor set $\mathcal{S}$, there is an instant $t \in I$ for which there is a consistent schedule G intact at t such that every activity $A \in \mathcal{S}$ has $A_e^G \leq t$ and every activity $B \notin \mathcal{S}$ has $t \leq B_s^G$.*

Proof: Note that $\max_{A \in S} A_{elb}$ in the theorem is an instant. □

Notice that $Avail_G(t)$ for an intact schedule at an instant $t \in I$ depends only on which activities are scheduled to end before t, so all the consistent intact schedules for a particular set constrained to occur before t will have the same availability at t. We next show that we can find $L_{max}(t)$ where t is an instant by only considering the intact schedules.

Lemma 1 *Suppose a schedule G' is derived from a schedule G by moving a producer activity earlier, or a consumer activity later (and no other changes). Then $Avail_G(t) \leq Avail_{G'}(t)$ for all t.*

Proof: Let A be the activity that is moved. We will just consider the producer case. (The consumer case is similar.) The contribution of A to the availability at t depends only on the fraction of A that precedes t. This can only stay the same or increase if A is moved earlier. □

Note that $Avail_G(t)$ is well-defined even if G is an inconsistent schedule, and the lemma still applies in that case.

Note also that a set of activities that all have the same start and end times may be treated as a single activity as far as the lemma is concerned provided they are moved as a unit. In this case, the set of activities may be considered to be a net producer if the rates of the individual activities sum to a non-negative quantity, and a net consumer otherwise.

Theorem 2 *Consider a consistent LRTN with integer temporal constraints and unit durations ($a_d = 1$). For any consistent schedule G, for all instants $t \in I$ there is a consistent schedule G' such that G' is intact at t and $Avail_G(t) \leq Avail_{G'}(t)$.*

Proof: Consider any consistent schedule G and instant $t \in I$. Let $t_0, \ldots, t_{m-1}$ be the activity start times in the open interval $(t-1, t)$ in increasing order, and set $t_m = t$. Let $G_0 = G$. We define a sequence of schedules $G_i, 1 \leq i \leq m$, which are not necessarily consistent.

To construct G_{i+1} from G_i:

1. Calculate $r_i = \sum_{A | A_s^{G_i} = t_i} a_r$.
2. If $r_i < 0$ then reassign all activities A starting at t_i to start at t_{i+1}. Otherwise, if $r_i \geq 0$ then reassign all activities A starting at t_i to start at $t-1$.

Since all activities have unit duration, by Lemma 1, each G_{i+1} has equal or higher availability at t than G_i.

We now show that each move preserves the activity ordering induced by G, i.e., if $A_s^{G_i} \leq B_s^{G_i}$ then $A_s^{G_{i+1}} \leq B_s^{G_{i+1}}$. If $A_s^{G_i} = B_s^{G_i}$, then if A and B are moved, they are moved together. If $A_s^{G_i} < B_s^{G_i}$ then the amount of any movement of A and B towards each other does not exceed their separation $B_s^{G_i} - A_s^{G_i}$ so the ordering is not changed. Thus, the (non-strict) start ordering is the same for all the G_i. The end time ordering is the same as the start time ordering.

Note that as a result of the movements, no activities in G_m start in the interior of the interval $[t-1, t]$. Thus, the last schedule G_m is intact at t with all activity orderings the same as they were in G. Note that since G is a consistent schedule, it respects the anti-precedence orderings, so G_m does also (even though it may not itself be consistent.) By construction, the set of activities scheduled to end prior to or at t in G_m determines a predecessor set whose net availability $Avail_{G_m}(t)$ equals or exceeds $Avail_G(t)$.

Also, note that if $A_s < t$ in G, then $A_{slb} \leq t-1$ (since the constraints, and hence the bounds, are integer-valued), and so $A_{elb} \leq t$ (because of unit durations). Similarly, if $B_s > t-1$ in G, then $t \leq B_{sub}$. Thus, Theorem 1 guarantees that there is a consistent schedule G' intact at t whose availability at t equals the net availability of the predecessor set. □

Corollary 2.1 *Consider a consistent LRTN with integer temporal constraints and unit durations ($a_d = 1$). Then for all instants $t \in I$, at least one schedule G justifying $L_{max}(t)$ is an intact at t schedule.*

Proof: Follows by details of construction of G from G'. □

We have seen that we can find $L_{max}(t)$ at the instants (elements of I) by only considering intact schedules. The significance of focusing on intact schedules is that the extended linear nature of the resource impact becomes irrelevant and we can use the maxflow approach to determine the optimum intact schedule. (In the LRTN case, the nodes in the flow graph correspond to the intact *activities* rather than the events as in the RTN framework.)

We next show that L_{max}^+ is achieved at an instant (element of I). A similar result applies to L_{min}^-. We remark that a local maximum may persist over an interval so achieving L_{max}^+ at an instant does not exclude achieving it at a non-instant and vice versa.

Theorem 3 *Consider a consistent LRTN with integer temporal constraints and unit durations ($a_d = 1$). Then L_{max}^+ is achieved at an instant.*

Proof: Consider any schedule G for which L_{max}^+ is achieved at some time t that is not an instant. There are two cases:

Case 1: Suppose no activities in G are split at t. Let $\mathcal{S}$ be the predecessor set consisting of all activities ending before or at t. Theorem 1 shows that there is an instant $t' \in I$ and a consistent schedule G' intact at t' such that all activities $A \in \mathcal{S}$ end before or at t' and all activities $B \notin \mathcal{S}$ are start at or after t'. Thus, $Avail_{G'}(t') = Avail_G(t) = L_{max}^+$.

Case 2: Suppose there is some activity split at t. Let $\mathcal{T}$ be the set of all such activities. We use a clearing technique similar to the proof of Theorem 2. Let $t_0, \ldots, t_{m-1}$ be the

activity start times in the open interval $(t-1, t)$ in increasing order, and set $t_m = t$. Let $G = G_0$. We define the sequence of schedules G_i as follows: To construct G_{i+1} from G_i:

1. Calculate $r_i = \sum_{A|A_s^G = t_i} a_r$.
2. If $r_i < 0$ then reassign all activities A starting at t_i to start at t_{i+1}. Otherwise, if $r_i \geq 0$ then reassign all activities A starting at t_i to start at $t - 1$.

Similar to the proof of Theorem 2, Lemma 1 guarantees that each G_{i+1} has higher or equal availability at t than G_i. Again, the G_i are not necessarily consistent schedules since the movements may violate some temporal constraints, but each move preserves the activity ordering induced by G, and so respects the anti-precedence relations. Thus, the set $\mathcal{S}$ of activities scheduled to end prior to or at t by the last schedule G_m, which satisfies $Avail_{G_m}(t) \geq Avail_G(t)$, is a predecessor set. Theorem 1 and its corollary then guarantees that there is a consistent schedule G' whose availability at some instant $t' \in I$ equals $Avail_{G_m}(t)$.

Since $Avail_G(t) = L^+_{max}$, the result follows.

□

Suppose we are given a consistent LRTN with integer temporal constraints and activity durations a_d consisting of integer constants. Suppose also that all activity rates a_r are specified. The following algorithm finds a bounding-above approximation of L_{max} and a bounding-below approximation of L_{min} that is tight at the instants:

1. Split each activity A into a_d unit activities and impose the necessary temporal constraints to preserve solutions. Note that splitting may increase the number of instants, but it will always be less than the horizon size h as a crude bound. [3] Compute the AllPairs shortest-path graph for the STN (to derive the anti-precedes relations).
2. At each integer instant $i \in I$, pose the following flow problem to find $L_{max}(t)$: denote the set of completely pending activities $P_{\mathcal{A}}(i)$ and let $A, B \in P_{\mathcal{A}}(i)$. If activity A anti-precedes B then the flow problem contains an arc $A \rightarrow B$ with infinite capacity. If $a_r > 0$ then the problem contains an arc $\sigma \rightarrow A$ with capacity a_r. If $a_r < 0$ then the problem contains an arc $A \rightarrow \tau$ with capacity $|a_r|$. To construct the flow problem to find $L_{min}(t)$, if $a_r > 0$ then the problem contains an arc $A \rightarrow \tau$ with capacity a_r, and if $a_r < 0$ then the problem contains an arc $\sigma \rightarrow A$ with capacity $|a_r|$.
3. As noted earlier, the set of reachable activities in the residual flow network is a predecessor set. $L_{max}(t)$ (resp. $L_{min}(t)$) is justified by the schedule ensuring all of these activities end before t. There is a consistent schedule satisfying this property by Theorem 1, and there is an intact schedule by Corollary 2.1.

The computational complexity of this algorithm is $O(n^3h^3)$ where h is the end horizon. If there are n activities in the LRTN, then at worst there are nh activities after transforming the LRTN in Step 1 of the algorithm. The AllPairs propagation using Floyd-Warshall or Johnson's Algorithm is thus $O(n^3h^3)$ (Cormen *et al.* 2001), and simple implementations of the Maximum Flow are $O(nm)$ where m is the number of edges in the flow graph. Using the incremental envelope calculation of (Muscettola 2004) reduces the complexity of all of the flow calculations to $O(n^3h^3)$.

[3] An alternative bound to h is $O(nD)$ where $D = max_{A \in \mathcal{A}} a_d$ is the largest original duration.

Finding L^-_{max} and L^+_{min} in Polynomial Time

The argument developed in the previous section does not apply to L^-_{max} or L^+_{min}. Consider a simple LRTN as shown in Figure 2. We see that L^-_{max} is determined by *two* schedules, one with decreasing availability and one with increasing availability, that intersect at $t = 1\frac{1}{2}$. Even more interesting, we see that L^-_{max} occurs at a time that is *not* an instant.

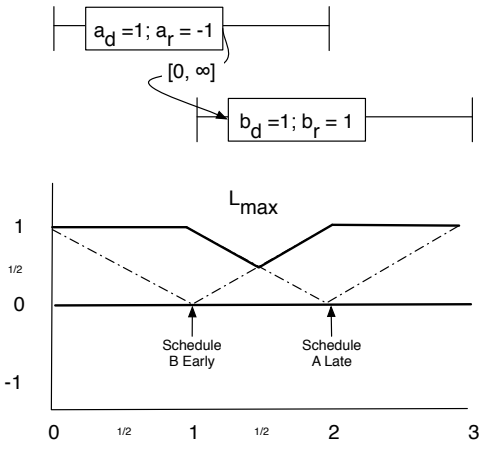

Figure 2: An LRTN for which L^-_{max} does not occur at an instant.

It is tempting to think that after finding the schedules justifying $L_{max}(t)$ at the instants, we might assemble L_{max} at intermediate times by projecting these schedules forwards and backwards and taking the intersections. A more complex LRTN is shown in Figure 3. In this case we see that there can be multiple slope changes between instants. We also note that the schedule justifying the envelope at times $1\frac{1}{3}$ to $1\frac{2}{3}$ does not justify the envelope at the instants.

While Theorem 3 shows us that we can compute L^+_{max} by finding $L_{max}(t)$ at h instants (and often closer to $O(n)$ instants), Figures 2 and 3 indicate that it may still be difficult to compute all of L_{max}. It is worth noting that finding L^+_{max} and L^-_{min} is sufficient to drive sound and complete search: if $L^-_{min} < 0$ or $L^+_{max} > R_{ub}$ and all timepoints are ordered or constrained to be identical, then the flaw is unresolvable, so we fail and must backtrack. Otherwise, we continue to make ordering decisions to resolve the flaws. However, finding L^-_{max} and L^+_{min} can eliminate fruitless branches early, even before all the ordering decisions have been made.

Local minima may also occur at instants such as the latest end times of consumers when no other activities are pending. However, the examples above show that some local minima of $L_{max}(t)$ may occur at non-integer times. We must account for all local minima of $L_{max}(t)$ by finding the times

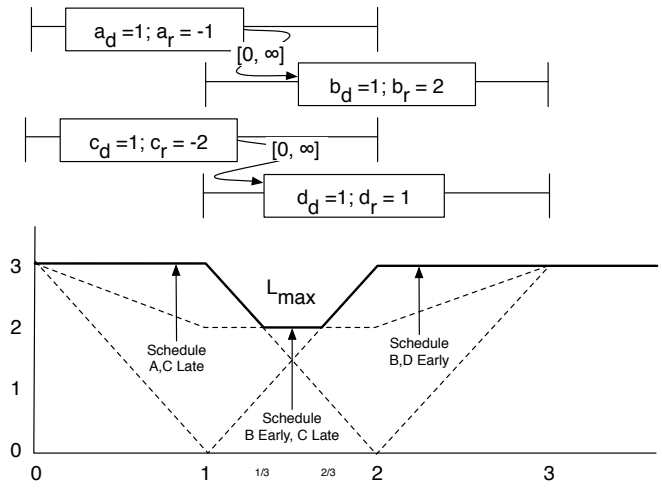

Figure 3: An LRTN with multiple slope changes between consecutive instants.

at which local minima can occur, and finding the envelope at these times. In order to do so, we need to extend the previous results we have proved so that they apply to times in between the instants (elements of I).

Consider an LRTN with integer temporal constraints and unit durations. Let λ be a real number such that $0 < \lambda < 1$. We can create a new LRTN by splitting each activity A into two activities, a *prefix* A^P of duration λ and a *suffix* A^S of duration $1 - \lambda$ and a temporal constraint ${A^S}_s = {A^P}_e$. We refer to A as the *parent* of A^S and A^P, and the new LRTN as a λ-*chopped* LRTN. We will denote the set of activities of the λ-chopped LRTN by $\mathcal{A}_\lambda$.

Lemma 2 *Consider an LRTN with integer temporal constraints and unit durations. Let $A, B \in \mathcal{A}_\lambda$. Then $d(B_e, A_s) < 0$ implies A anti-precedes B (i.e., $d(B_e, A_e) \leq 0$). Thus, either A anti-precedes B or B can follow A.*

Proof: Let the parents of A and B be A' and B', respectively. We distinguish 3 cases:

1. A and B are both prefixes or A and B are both suffixes. Then $d({B'}_e, {A'}_s) < 0$, so A' anti-precedes B', and hence A anti-precedes B.
2. A is a prefix and B is a suffix. Then $A_s = {A'}_s$ and $B_e = {B'}_e$. Thus, $d({B'}_e, {A'}_s) < 0$, so A' anti-precedes B', and hence A anti-precedes B.
3. A is a suffix and B is a prefix. Then either B' can come after A' in which case B can come after A, or A' anti-precedes B'. In the latter case, B (prefix of B') must end before A (suffix of B') starts, so A anti-precedes B.

□

Theorem 4 *Consider an LRTN with integer temporal constraints and unit durations, and suppose $\mathcal{S} \subset \mathcal{A}_\lambda$ is a predecessor set in its λ-chopped LRTN, for any $0 \leq \lambda \leq 1$. Let t be any time such that $max_{A \in \mathcal{S}} A_{e,lb} \leq t \leq min_{B \notin \mathcal{S}} B_{s,ub}$. Then there is a consistent schedule G such that G is intact at t with respect to the λ-chopped activities, and such that every activity in $A \in \mathcal{S}$ has $A_e^G \leq t$ and every activity $B \notin \mathcal{S}$ has $t \leq B_s^G$.*

Proof: This is a generalization of Theorem 1. The same proof technique suffices using Lemma 2, which is a generalization of the property 4 preceding Theorem 1. □

Theorem 5 *Consider an LRTN with integer temporal constraints and unit durations. For any consistent schedule G and for all times t, there is a consistent schedule G' such that $Avail_G(t) \leq Avail_{G'}(t)$ and such that G' is intact at t with respect to the λ-chopped LRTN where $\lambda = t - \lfloor t \rfloor$.*

Proof: We use a more elaborate version of the clearing technique of Theorem 2. In this case we clear activities that are split at either $\lfloor t \rfloor$ or $\lceil t \rceil$. The result of this will be that the moved activities ultimately end up starting at $\lfloor t \rfloor - 1$ or $\lfloor t \rfloor$ or $\lceil t \rceil$.

Consider any schedule G and time t. Since Theorem 2 applies if t is an instant, assume t is not an instant. Let t_i be the activity start times in the closed interval $[\lfloor t \rfloor - 1, \lceil t \rceil]$ in increasing order. (Note the interval spans two units if t is a non-integer.) We include $\lfloor t \rfloor$ and $\lceil t \rceil$ in the sequence even if there are no activities starting at these times. Suppose $t_0, \ldots, t_m$ are the elements in the sequence. Let $G_0 = G$. We define a sequence of schedules G_i (not necessarily consistent) as follows. To construct G_{i+1} from G_i:

1. If t_i is an integer (e.g.,$\lfloor t \rfloor$), do nothing (i.e., $G_{i+1} = G_i$).
2. Otherwise, calculate $r_i = \sum_{A | A_s^{G_i} = t_i} a_r$. If $r_i \geq 0$ then reassign all activities A starting at t_i to start at $\lfloor t_i \rfloor$. (Note that $\lfloor t_i \rfloor$ may be either $\lfloor t \rfloor - 1$ or $\lfloor t \rfloor$ depending on the value of t_i.) Otherwise, if $r_i < 0$ then reassign all activities A starting at t_i to start at t_{i+1}.

As before, all activities have unit duration, and Lemma 1 guarantees each G_{i+1} has higher or equal availability at t than G_i. The argument that activity ordering is preserved is identical to Theorem 2.

Note that the G_i are not necessarily consistent schedules in that they may violate binary temporal constraints between activities. However, absolute time windows on activity start times and end times are protected. To see this, note that activities starting at a time x are ultimately moved either to $\lfloor x \rfloor$ or $\lceil x \rceil$. Since the constraints and hence the time bounds are integer-valued, if x is within the bounds then both these values are also within the bounds.

As a result of the movements, no activities are split at $\lfloor t \rfloor$ or $\lceil t \rceil$ in G_m. If t is a non-integer, activities may be split at t itself in G_m, but only if they start at $\lfloor t \rfloor$ (otherwise they would cross $\lfloor t \rfloor$ or $\lceil t \rceil$). Note that since G is a consistent schedule, it respects the anti-precedence orderings, so G_m does also (even though it may not itself be consistent).

We now form the λ-chopped LRTN where $\lambda = t - \lfloor t \rfloor$; thus every A^P has duration $t - \lfloor t \rfloor$ and every A^S has duration $\lceil t \rceil - t$. Let $\mathcal{S}$ be the set of λ-chopped activities A such that $A_s^{G_m} \leq t$; we observe that $\mathcal{S}$ is a predecessor set in the λ-chopped LRTN.

We next show that t lies in the "sweet spot" $\max_{A \in \mathcal{S}} A_{elb} \leq t \leq \min_{B \notin \mathcal{S}} B_{sub}$ for $\mathcal{S}$ required by The-

orem 4. This follows from the fact noted above that absolute time windows on activity start times and end times are protected in the G_i. Thus, if a chopped activity A ends up before t (i.e., in $\mathcal{S}$), then we must have $A_{elb} \leq t$. Similarly, if B ends up beyond t then $t \leq B_{sub}$.

Thus, we can apply Theorem 4 to conclude that there is a consistent schedule G' such that t separates the activities in λ from those not in λ. This will have the same availability at t as G_m. Therefore, it equals or exceeds the availability of the original schedule G_0 at t. □

Theorem 5 justifies using the maximum flow formulation described previously to find (λ-chopped) predecessor sets that identify $L_{max}(t)$ at non-integer times t. However, we don't want to survey a large number of times t between a pair of instants $[i, i+1]$. Furthermore, it is an open question whether there may be an exponential number of changes of slope of $L_{max}(t)$ in the interval $[i, i+1]$.

We avoid this problem by showing that we can find the time t at which L_{max} achieves a local minimum between a pair of consecutive integers and the value of the local minimum at the same time by formulating this problem as a linear program. Observe that every flow problem defined by a chopped LRTN between a pair of integers $i, i+1$ is *identical* except for the capacities of the pipes from the source or to the sink, which may vary linearly. We exploit a result proved in (Muscettola 2002), where the *incremental* availability is defined as the contribution to $L_{max}(t)$ from the events pending at t, i.e., those represented in the flow problem.

Theorem 6 *Consider the maximum flow problem F derived from a maximum availability problem at some time t. Then the incremental availability at t equals the residual capacity of the source pipes of F in a maximum flow.*

Proof: See (Muscettola 2002); this is an easy consequence of the methods of that paper, and is essentially stated as an intermediate result within the proof of Theorem 1, p. 149. [4] □

The significance of this theorem is that the increment in resource availability due to the pending activities is equal to the residual of the source pipes in the maximum flow solution. Note that "Flow = Capacity − Residual." Thus, maximizing the flow at t is equivalent to minimizing the residual at t. This means that the minimum over t of the maximum incremental availability at t can be found by minimizing over t the minimum residual (over all flows) at t. This suggests we could find $\min_t(r_{flow}(\mathcal{A}^+_\lambda))$ by simply minimizing $r_{flow}(\mathcal{A}^+_\lambda)$ (i.e., the residual of $\mathcal{A}^+_\lambda$ in an arbitrary flow *flow*) and letting t vary along with the flow. Since the constraints in a flow network are linear in both the flows and the capacities, we can build a linear program to do this.

The relevant flow network is obtained by considering the λ-chopped activities (where $\lambda = t - \lfloor t \rfloor$) that are pending at t and their anti-precedence relations, and inserting pipes as discussed previously. Notice that the capacities of the source and sink pipes will vary linearly with λ since the durations of the chopped activities vary linearly with λ.

[4] What the argument boils down to is that input flow must equal output flow, so for the residual set (where the output capacity is fully used), the unused input capacity must equal the difference of the input and output capacities.

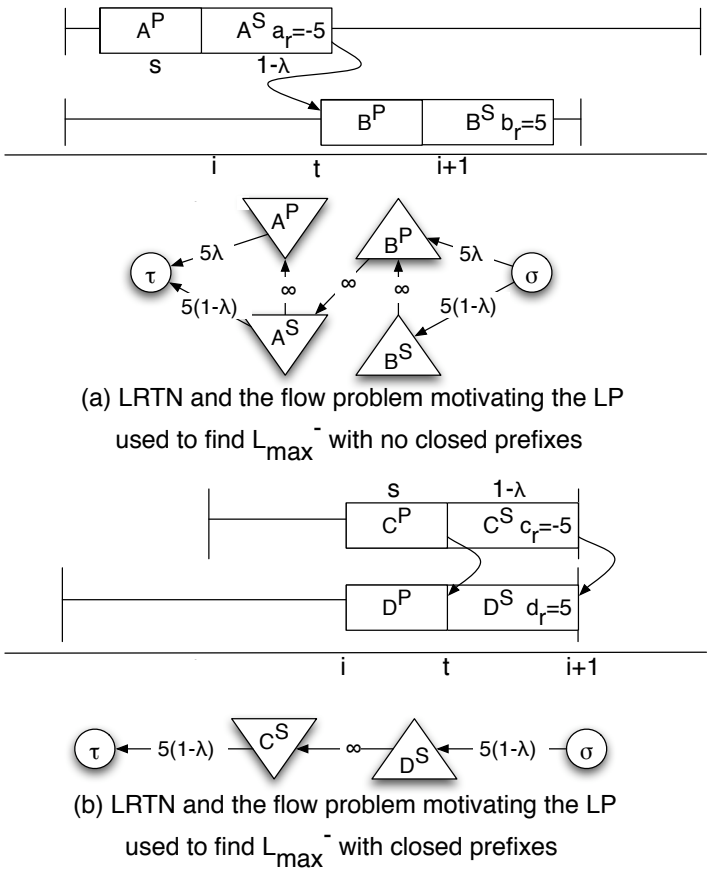

Figure 4: The flow problems motivating the LP used to find L^-_{max}.

However, we actually want to minimize over λ the *total* availability, which is obtained by adding the availability from the *closed* activities to the incremental availability. Some of the λ-chopped activities will be in the closed set. Their durations, and hence their resource contributions, also vary linearly with λ. Thus, we must account for any part of an activity (in the non-λ-chopped LRTN) that must be completed by $t = i + \lambda$, since these also contribute to $L_{max}(t)$. This is the case for a prefix A^P such that $A_{eub} = i + 1$; it must contribute $a_r\lambda$ to the objective of the linear program, even though the prefix does not appear in the flow problem because it is in the set of closed activities. Examples of these flows are shown in Figure 4.

We create an LP consisting of variables representing capacity in the pipes, plus one additional variable, λ, and the usual flow conservation and capacity constraints. The capacity constraints are all parameterized by λ. We constrain $0 \leq \lambda \leq 1$. We minimize the residual (the excess capacity of the source pipes) plus the required contributions of all activities that must end at $i + 1$: thus, we minimize $r_{flow}(\mathcal{A}^+_\lambda) + \sum_{A \in \mathcal{A} | A_{eub} = i+1} a_r\lambda$.

The solution to this LP is, by construction, a time $t =$

$i + \lambda$ at which the residual is minimized, and the value of the residual; Theorem 6 shows us how to calculate $L_{max}(t)$ from this residual. Since the solution minimizes the residual at t (over all flows at t), it maximizes the flow at t; thus by Theorem 5, it provides a schedule that justifies $L_{max}(t)$. Thus, the LP solution gives us the local minimum of L_{max} in the interval $[i, i + 1]$.

To find L^-_{max} requires solving at most h LPs derived from flow problems on the chopped LRTNs. The flow problems have at most $2nh + 2$ nodes and therefore at most $4h^2n^2+2nh$ pipes. The LPs, then, have at most $4n^2h^2+2nh$ variables. This leads to a worst-case complexity (assuming Karmarkar's algorithm) of $O(n^6h^6)$ to solve each LP, for a total complexity of $O(n^6h^7)$ to find L^-_{max}. To find L^+_{min}, we use a symmetrical approach.

In closing, we observe that the LP construction shows that L_{max} must be concave between instants; thus, there can be no local maxima between a pair of instants. We therefore can construct the entire envelope.

Conclusions and Future Work

We have generalized Resource Temporal Networks to Linear Resource Temporal Networks (LRTNs). We showed how to construct tight bounds for resources in LRTNs in polynomial time in the horizon h and the number of activities n, assuming metric temporal constraint data consists exclusively of integers.

While this work constitutes proof that a precise characterization of the resource availability for networks of activities with linear resource impact can be constructed in polynomial time, the algorithm described here may not be of immediate practical use because of the potentially large number of activities resulting from the chopping to unit durations. We believe the approach described here can be improved considerably by splitting activities only as needed in order to guarantee the existence of a dominating intact schedule. A formal analysis may eliminate the dependence of the complexity on the time horizon h. A side benefit of this would be to make more plausible claims that the approach can be applied to problems with rational data in temporal constraints; extensions to real numbers in temporal constraints may be a more difficult goal to achieve. A large source of complexity is the LP method used to compute the local minima, which may be more costly than maxflow algorithms. This suggests it may be fruitful to seek a maxflow-like algorithm that would directly solve a parameterized flow problem with linearly varying capacities.

In this paper we have assumed rates and activity duration are provided as inputs. Allowing rates or durations to vary requires further work. In this context, the class of LRTNs may need to be adapted so that plan quality is an explicit function of activity duration; this would provide motivation for a scheduler to search over activity duration.

More work is needed to analyze the best approach for solving search problems. The approach described in (Frank 2004) enables more pruning of the search than that of (Muscettola 2002), at higher cost. Presumably the same approach could be of benefit in the context of LRTNs, but at still higher cost. A tight bound is not necessarily the best to use in search. When solving scheduling problems, it is worth comparing approaches that employ the formulation of (Muscettola 2002) to the formulation suggested here, even when the true problem contains activities with linear resource impact. While our formulation is precise, the complexity difference may ultimately make the cheaper approach more tolerable. Finally, a study on the impact of the different approaches on schedule flexibility, as is done in (Policella *et al.* 2004), is also worthwhile.

References

Ahuja, R.; Magnanti, T.; and Orlin, J. 1993. *Network Flows*. Prentice Hall.

Cheng, C., and Smith, S. 1995. A constraint posting framework for scheduling under complex constraints. In *Joint IEEE/INRIA conference on Emerging Technologies for Factory Automation*.

Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. 2001. *Introduction to Algorithms*. MIT Press.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–94.

Frank, J. 2004. Bounding the resource availability of partially ordered events with constant resource impact. In *Proceedings of the* 10^{th} *International Conference on the Principles and Practices of Constraint Programming*.

Jónsson, A.; Morris, P.; Muscettola, N.; Rajan, K.; and Smith, B. 2000. Planning in interplanetary space: Theory and practice. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*.

Laborie, P. 2003a. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143:151–188.

Laborie, P. 2003b. Resource temporal networks: Definition and complexity. In *Proceedings of the* 18^{th} *International Joint Conference on Artificial Intelligence*, 948 – 953.

Morris, P.; Muscettola, N.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proceedings of the* 15^{th} *National Conference on Artificial Intelligence*.

Muscettola, N. 2002. Computing the envelope for stepwise constant resource allocations. In *Proceedings of the* 8^{th} *International Conference on the Principles and Practices of Constraint Programming*.

Muscettola, N. 2004. Incremental maximum flows for fast envelope computation. In *Proceedings of the* 14^{th} *International Conference on Automated Planning and Scheduling*.

Policella, N.; Smith, S.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *Proceedings of the* 14^{th} *International Conference on Automated Planning and Scheduling*.